Complexity Signatures for System Health Monitoring

Kagan Tumer and Adrian Agogino
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
650-604-4940
ktumer@mail.arc.nasa.gov, adrian@email.arc.nasa.gov

Abstract—The ability to assess risk in complex systems is one of the fundamental challenges facing the aerospace industry in general, and NASA in particular. First, such an ability allows for quantifiable trade-offs during the design stage of a mission. Second, it allows the monitoring of the health of the system while in operation. Because many of the difficulties in complex systems arise from the interactions among the subsystems, system health monitoring cannot solely focus on the health of those subsystems. Instead system level signatures that encapsulate the complex system interactions are needed. In this work, we present the Entropy-Scale (ES) and Entropy-Resolution (ER) system-level signatures, that are both computationally tractable and encapsulate many of the salient characteristics of a system. These signatures are based on the change of entropy as a system is observed across different resolutions and scales.

We demonstrate the use of the ES and ER signatures on artificial data streams and simple dynamical systems and show that they allow the unambiguous clustering of many types of systems, and therefore are good indicators of system health. We then show how these signatures can be applied to graphical data as well as data strings by using a simple "graph-walking" method. This methods extracts a data stream from a graphical system representation (e.g., fault tree, software call graph) that conserves the properties of the graph. Finally we apply these signatures to analysis of software packages, and show that they provide significantly better correlation with risk markers than many standard metrics. These results indicate that proper system level signatures, coupled with detailed component-level analysis will enable the automatic detection of potentially hazardous subsystem interactions in complex systems before they lead to system deterioration or failures.

TABLE OF CONTENTS

0-7803-7231-X/01/$10.00/
IEEEAC paper # XXX

1. INTRODUCTION

In large complex system composed of many interacting components determining the health of the full system is significantly more involved than simply assessing the health of all the subsystems. The unpredicted interactions among those subsystems are often the leading causes of system deterioration or outright failure [1]. However, because of the sheer number of variables involved, accurately modeling those interactions is difficult at best and impossible at worst.

It is therefore crucial to obtain methods that characterize complex engineered systems in a way that will both allow better design trade-offs during the design stages and provide detailed health monitoring signals during operation. Though an accurate measure of system complexity would be a significant step in that direction, what is of more immediate concern is a establishing a connection between the salient characteristics of the system and its inherent risk. In this work we focus on providing system level signatures that are good markers for the crucial system behaviors and therefore are good predictors of risk, rather than grapple with a precise definition of system complexity. Indeed the definition of a broadly applicable complexity measure has proven to be both elusive and in many cases contentious [11].

A survey on complex system science including a discussion of the many proposed complexity measures is presented in [11]. Among such measures, the Kolmogorov (or algorithmic) complexity [8] is historically the most studied measure, while Self-Dissimilarity is one of the more novel and interesting measures [16]. The Kolmogorov complexity, defined as the minimum number of bits into which a string can be compressed by a Universal Turing machine without loss of information, is deeply rooted in mathematical theory. It is often intuitively defined as length of the code that would generate the observed data. Though theoretically well founded, the Kolmogorov complexity is of little use in practice because it equates randomness with complexity and more importantly, it cannot be computed efficiently [11]. Self-Dissimilarity on the other hand is a promising empirical measure of complexity for a broad class of systems including naturally occurring and engineered ones [16]. It is based on the intuitive notion that complex systems process information differently at different scales which leads them to exhibit rich and varied behavior when viewed across different scales. It has successfully been applied to clustering satellite images and has

been shown to discriminate chaotic and non-chaotic behavior in logistic maps [16]. However, it can be computationally expensive in some domains and to date, has not been been used to try to discriminate between healthy and non-healthy engineered systems. Other recent complexity measures include information or entropy based methods [6], [13], heuristic based methods [2], and methods based on stochastic complexity [12].

The system-level signatures we present in this work encapsulate the interactions among the system components, but remain computationally tractable for statistical analysis. In particular, this method is aimed at both numerical and graphical data: Numerical data in this context is any "raw" data that can be collected from sensors (e.g., images, streams from telemetry, bit strings) as well as "flattened" graphical data. Graphical data is higher-level representation of a system in the form of a flowchart or graph (e.g., fault tree, design diagram, software call graph, schematics). In general, the use of simple statistics on either set of data yields little clue on the complexity of the system: Graphical data representations are too "abstract", while low-level data is too high dimensional to provide the required information about overall system health. Data reduction methods offer a potential solution for the latter, but because the properties pertinent to system health are often hidden in secondary system interactions, such methods do not generally provide good system level signatures.

The proposed system level signatures are based on Shannon entropy, which intuitively provides a measure of the uncertainty remaining in the system after an observation has been made [4]. Though a valuable statistical measure, entropy by itself is a poor predictor of complexity, because it treats randomness and complexity as the same: the system with the highest entropy is a system that is purely random. However, such a system is generally not considered to be of "high complexity". Furthermore, entropy only focuses on the frequency of patterns, not their ordering. As such a periodic binary string and a random binary string containing the same number of ones and zeros have the same entropy. To overcome these limitations of entropy as a system level signature we focus on the change of entropy as the system is viewed across different resolutions and scales. Though the distribution of ones and zeros may be the same in a random string and a periodic string, the distribution of two digit strings and three digit strings may be different. Finally, we extend this signature to graphical data by performing a walk on the data that results in numerical data, while retaining the subsystem interactions of the numerical data. Our results show that the Entropy-Scale (ES) and Entropy-Resolution (ER) signatures provide significantly more information about a system's behavior and therefore are good predictors of system health.

In Section 2 we use data networks to provide a motivating example of the impact of selecting the proper system signatures. We show how the health of the system is obscured by selecting natural or traditional system signatures. We then present domain specific alternatives that allow for more successful risk assessment. In Section 3 we discuss how to generalize this concept, and introduce the entropy based system level signatures. In Section 4 we illustrates how these signatures can be applied to data strings. In Section 5, we apply these signatures to software packages and show that the signatures are better indicators of system level risk (evaluated as the number of revisions) than traditional software complexity metrics. Finally, in Section 6, we highlight the implications of these results, discuss the limitations of the proposed method and suggest future research directions.

2. MOTIVATION: SIGNATURE SELECTION

This section shows the impact of selecting system level signatures in predicting the robustness of a set of simple data network models. The networks include include ring, small-world, hub, tree and grid topologies (the last three are shown in Figure 1). In this motivating example, the robustness of the network is determined as the ability of the network to withstand having nodes removed and still have all of the nodes of the network be reachable from any other node. In this instance, the risk associated with these networks is that communication between nodes is interrupted after a random set of nodes are removed. Furthermore, these networks are classified as "high risk" or "low risk" based on this metric. The question in this case is to determine good system level signatures that will readily distinguish between the two types of risk for various network topologies and sizes.

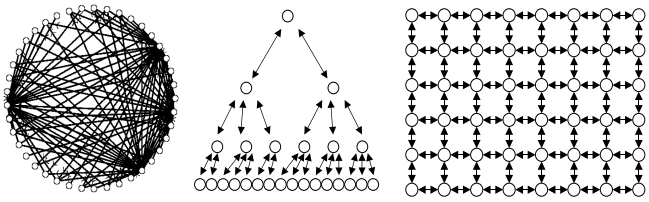
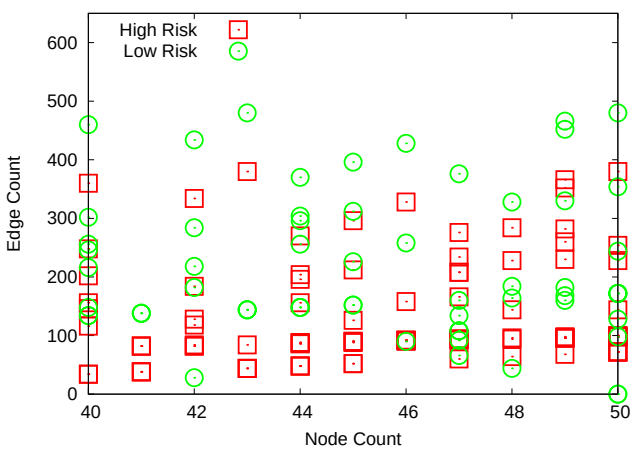


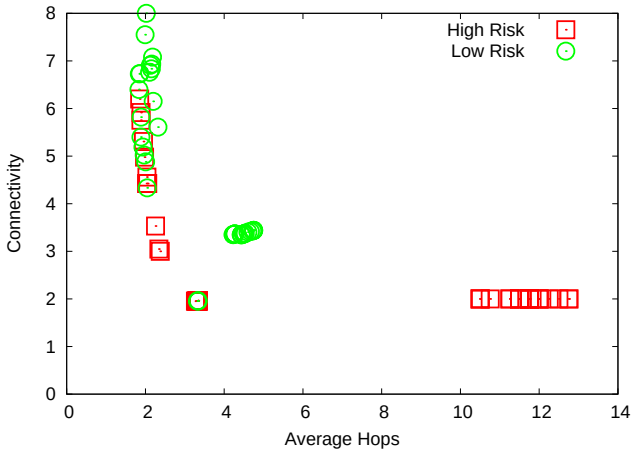
Figure 1. Different network topologies. Different system structures may look similar if inappropriate features or not enough features are used. A broad feature signature can alleviate this problem.

The two most obvious features to monitor in a network are the number of nodes and edges in the network. These two features form a "signature" as shown in Figure 2(a). Intuitively, one might expect certain network types as defined by this signature (e.g., networks with many nodes and relatively few edges) to have higher risk than other networks. However, Figure 2(a) shows that in the absence of additional information (e.g., network topology), this signature has little value in predicting network risk. This signature fails to provide discriminating features because it does not capture summary information about the topology of the network. Without a signature that captures how the nodes interact, the low risk (green circle) vs. high risk (red square) classification is nearly impossible.

Let us now investigate a new signature, one based on the con-



(a) Distribution of High-Risk/Low-Risk networks with respect to number of edges and nodes.



(b) Distribution of High Risk/Low Risk networks with respect to average connectivity and average number of hops.

Figure 2. Discriminating capability of two different sets of signatures. Red squares are high risk networks and green circles are low risk networks.

nectivity of the network (average number of neighbors for each node) and the network traversal count (average number of hops from one node to another). This signature has the exact same dimensionality as the edge/node count signature (e.g., both require two numbers) and can be visualized in an identical matter (e.g, 2-D graph). Figure 2(b) provides this signature for the same set of networks as in Figure 2(a). Note that in this instance, the classification of high risk and low risk networks is far easier in many parts of the signature space.

In this case, we used our knowledge of network topology to derive a signature (connectivity vs. average hop chart) that provided discriminating features for assessing system risk (network breakdown). In the following sections we generalize this idea by providing signatures that are good risk discriminators in large systems where the interactions among subsystems are sufficiently complex to preclude hand picking specific variables as indicators of system behavior.

In Figure 3 we present alternative signatures that assume no prior knowledge about the domain, such as network structure, importance of connectivity or traversal distance. Note, we provide these signatures here as a motivating example based on the networks discussed above. The details on how these signatures are obtained is discussed in Section 3. The signature in Figure 3(b) separates the networks into three clusters: low risk, high risk and undetermined. This is similar to the clustering in Figure 2(b) which also had an "undetermined" class of networks in the top left corner. Figure 3(a) separates the networks even more successfully into low and high risk classes. Also note that Entropy alone (e.g., scale 1 in ES signature) does not provide discriminating power, highlighting the need to view the full signature.

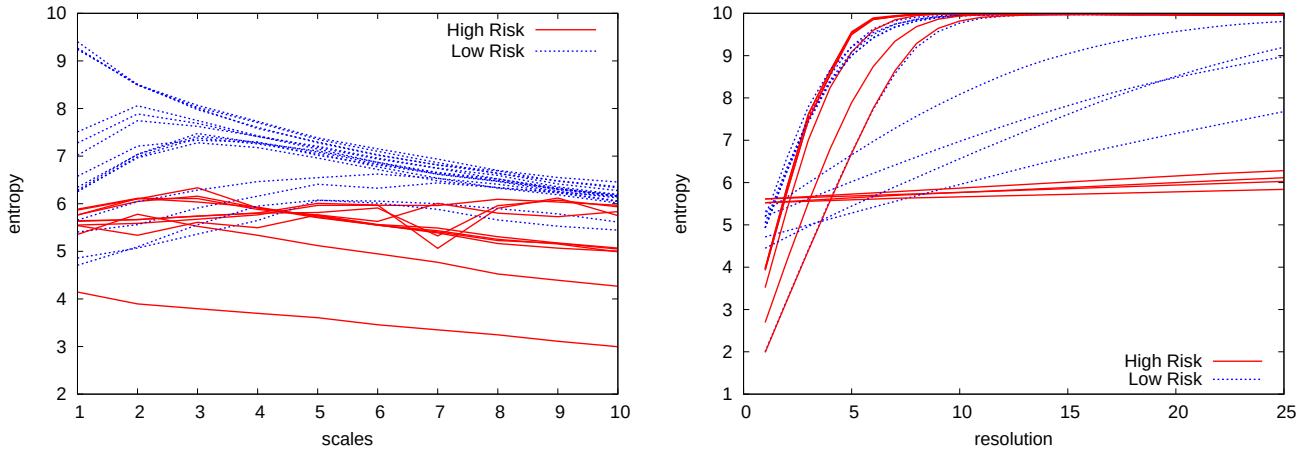
3. System Level Signatures

In this section we introduce two system level signatures based on Shannon Entropy. We first briefly present the concept entropy, then discuss the impact of changing the resolution and scale with which we study that system on entropy. For a given data string S of alphabet size A, where the probability of observing each alphabet element in that string is $p(s_i)$, the Shannon entropy is defined as:

$$H(S) = \sum_{i=1}^{A} p(s_i) log_2\ p(s_i) \quad (1)$$

Intuitively, the Shannon entropy provides the amount of uncertainty removed by observing the string. As such it has three main drawbacks as a system level signature related to system complexity. The first of these is that entropy is a measure of disorder, not complexity and therefore is maximal for purely random patterns. It provides the amount of order or disorder in the system, which may or may not be correlated with the complexity of the system. Indeed, many intuitive notions of complexity hinge on having subtle interactions among highly unpredictable components.

A second issue with using entropy as a complexity measure is that because it is based on the distribution of samples, it disregards the order in which the samples are observed. This property has the effect or removing potentially highly relevant information. Consider the following three binary strings: s_1, a string of alternating zeros and ones; s_2, a random binary string where $p(0) = p(1) = .5$; and s_3, a string of double ones alternating with double zeros. Because each string has the same distribution of ones and zeros, each string has the same Shannon entropy. Clearly fundamental properties of the



(a) Entropy-Scale Signature (resolution $r = 3$).

(b) Entropy-Resolution Signature (at scale 1).

Figure 3. Entropy based signatures for risk assessment in different network types. Solid (red) lines are high risk networks; dashed (blue) lines are low risk networks.

system have not been captured by the entropy, and representing the systems that generated those strings simply by their entropy will not be of much use.

The third issue with using Entropy as complexity measure is that entropy ignores the sample labels. For binary strings this is not a major problem, but for strings with large alphabets, or for strings with real numbers, this can be crucial: the concept of proximity is lost. Because of these deficiencies, Shannon entropy unsuitable to be used directly as a complexity measure. In this work we focus on signatures that directly deal with the first two issues. In Section 6, we discuss an extension that also addresses the third issue.

The signatures we propose are based on the change in Entropy when the string is viewed at different resolutions and at different scales. In this context, resolution means the size of the unit elements that constitute the data. We can analyze an image at a single pixel level, at 2x2 (or nxn) windows, or we can analyze a book by looking at the letters, words or sentences. Each of those choices will provide analysis at a different resolution. Scaling on the other hand, corresponds to zooming in or out of the data.

Entropy-Resolution Signature

The Entropy-Resolution signature overcomes the first two deficiencies described above. The intuitive idea behind looking at entropic variability is that the change of entropy will provide insight into the system. A random pattern will have high entropy, but it will be high regardless of the resolution at which it is observed. For example, a random string of letters across a page will not appear more or less random when viewed at a letter resolution, a word resolution or a sentence resolution. Therefore, the entropy variation across resolution will be a straight line.

As an example, let us return to the three binary strings again: The entropy of s_2 will be the maximal entropy at each resolution as each pattern is equally likely until the resolution is high enough to introduce sampling effects. At resolution $r = 2$, the entropy of s_1 however will be identical to its entropy at resolution $r = 1$, since only two of the possible four strings (01 and 10) are observed. At resolution $r = 2$, the entropy of s_3 will also be maximal since all possible two bit strings will occur with equal frequency. So for resolution $r = 2$, we will have:

$$H_{s_2}^{r_2} = H_{s_3}^{r_2} > H_{s_1}^{r_2} \quad (2)$$

where $H_{s_i}^{r_j}$ is the entropy of the ith string at resolution j. At resolution $r = 3$, s_2 and s_3 also separate, since s_3 will only have certain three-bit samples (it will only have 110, 001, 011, 100). For resolution $r = 3$, we will have:

$$H_{s_2}^{r_3} > H_{s_3}^{r_3} > H_{s_1}^{r_3} \quad (3)$$

The ER signature separates the three strings by providing different entropy values at different resolutions. Intuitively, ordered or purely random systems will have simple (linear) ER signatures whereas more complex systems will exhibit a richer set of patterns.

Entropy-Scale Signature

The Entropy-Scale signature shares many of the characteristics of the ER signature. Scaling can be viewed as zooming in to or out of the data. Looking at the entropy across different scales will provide high level information as well as finer details (shape of a table, vs. grain in the wood). If the data set is well understood, the appropriate scale at which to analyze it may be obvious. If not, it has to be determined through

the ES signatures. However, even if the appropriate scale is known, looking at the full scale entropy signature provides a wealth of information about a system's health.

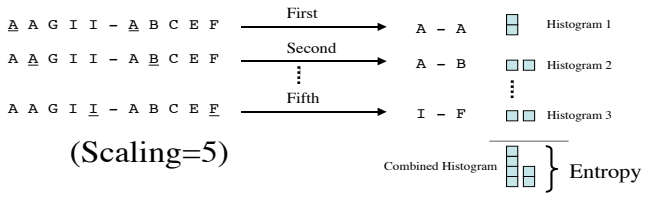


Figure 4. Scaling Symbolic Data: For a scale of five five data strings are formed by sampling every fifth element of the original data string using five different offsets from the beginning of the stream. A histogram is made of each stream, sorted and then combined to form one aggregated histogram.

For a scaling of size n we want to represent a set of n values in the data stream with a single value. While scaling an image is relatively straight-forward [1], scaling arbitrary streams of data is not. Techniques that work for images, such as averaging pixel values do not work data streams composed of arbitrary symbols. Even for data sets composed of scalar values, averaging can be tremendously influenced by how the data is processed. Instead of averaging this paper approaches scaling by looking at the distribution of the symbols as shown in Figure 4. What this method amounts to is decimating the data at the rate of the scale n. However, in order to limit the impact of starting points and random fluctuations, the operation is conducted n times.

For a scaling of size n, substrings are produced by sampling every n symbols in the original data-string. n substring are produced, each being sampled with a different offset from the beginning of the string. For example, a scaling of $n = 2$ for the data string 101010 would produce the substring 111 and 000. The histogram of each substring is then computed and sorted using an arbitrary sorting order. Then the sorted histograms of all of the substrings are added together to form a single histogram corresponding the the scaled view of the data. Distribution based measures, such as entropy, can then be used on the combined histogram to produce a signature value for the scaling. In the above string of alternating ones and zeros, we get an entropy of zero at scale 2 since both substrings are constant.

4. Signature Examples on Strings

In this section we show how signatures composed of different scales and resolutions capture the salient properties of three sets of binary strings. The first set consists of two simple repeating patterns. The second set consists of two simple repeating patterns of ascii characters. The third set consists of more complex patterns generated from cellular automata. Finally, we present a mixed string example where we blend different strings together. All the strings are ten thousand bits long. In all of the plots with varying scales the resolution is set to eight. In all of the plots with varying resolutions the scale is set to one.

[1] Even for images there are many scaling techniques such as box, triangle, b-spline and bicubic scaling. Each method has its own tradeoffs.

Simple Binary Strings

In this example two different binary strings are used. The first string is simply alternating ones and zeros: "10101010..." The second string is composed of sets of six consecutive ones separated by either one, two or three zeros (with an average of two zeros): "1111110111111000111111001111110 1..." Also from these patterns, two more patterns were generated by adding 10% noise from a uniform distribution of ones and zeros.

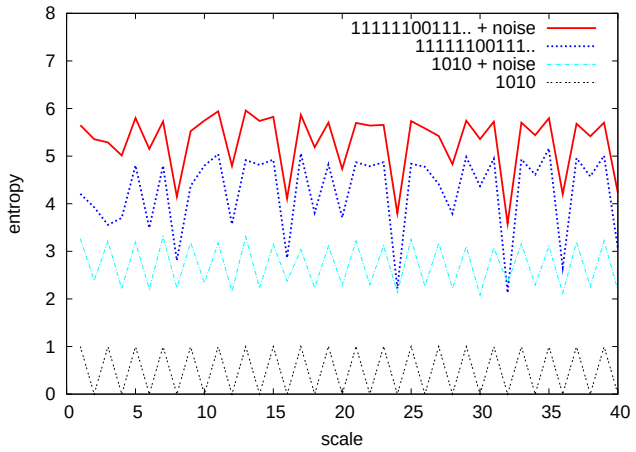
Figure 5 shows the ER and ES signatures of these strings formed from different scales. The signature for the first string is as simple as the string itself, as it alternates at every scale. When noise is added to the strings the signature looks very similar except that the noise causes it to have more entropy on average. The signature for the second string shows the more unpredictable nature of the string. The signature reflects that there are many patterns in this string. However, there is still a dominant pattern represented in the pattern as spikes in the signature at scales that are multiples of eight. This corresponds to the eight-bit pattern composed of six ones followed by an average of two zeros. When noise is added to this signature, its average entropy rises too, but the shape of the signature is still intact.

Eight-bit strings

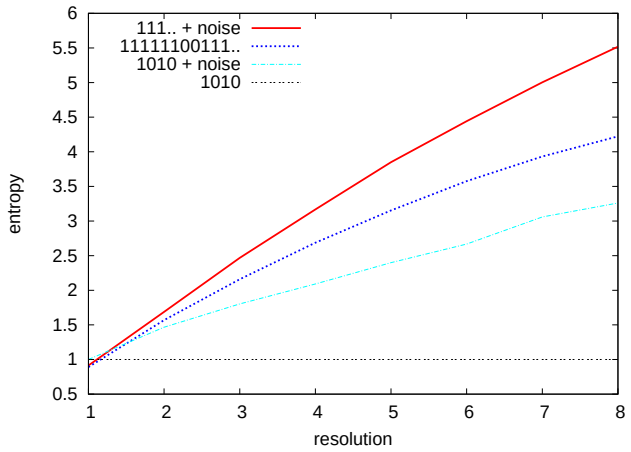
In this example two sets of strings are generated from the eight bit representations of ascii characters. The first string is composed of a repeated string of the ascii character 'A': "AAAAA..." The first string is composed of a repeated string of the ascii characters 'AZ' : "AZAZAZ..." Figure 6 shows the ES and ER signatures. Note the ES signature for the first pattern spiking at multiples of eight, which corresponds to the eight-bit length of the pattern. This signature also shows that for the second string there are spikes at multiples of eight corresponding to the length of the ascii characters. It also shows an even bigger spike at multiples at sixteen that is not present in the first string. This spike corresponds to the pattern being composed of two eight bit characters. Also note that like the previous strings, the addition of random noise increases the entropy uniformly, but does not significantly change the shape of the signature.

Cellular Automata

In this section we apply the ES and ER signatures to eight bit Cellular Automata (CA) [5], [15]. CAs are discrete dynamical systems that can be represented in simple 2D plots where the axis are space and time. The evolution of the system is prescribed by a set of rules, and at each step, each cell determines its next value as a function of the current states of other cells. Figure 7 shows six eight-bit CAs (evolving vertically).

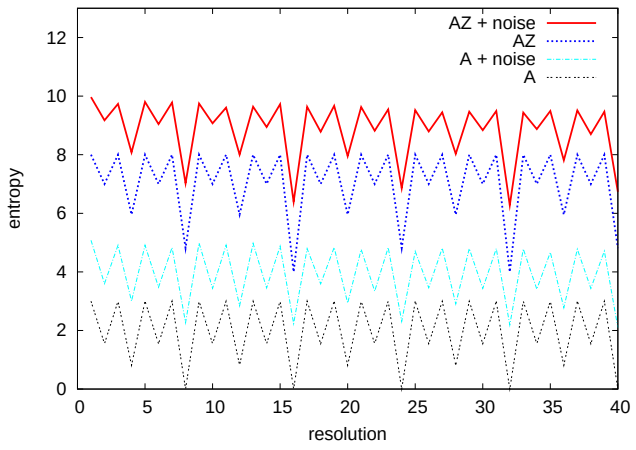


(a) Entropy-Scale signature (resolution $r = 8$).

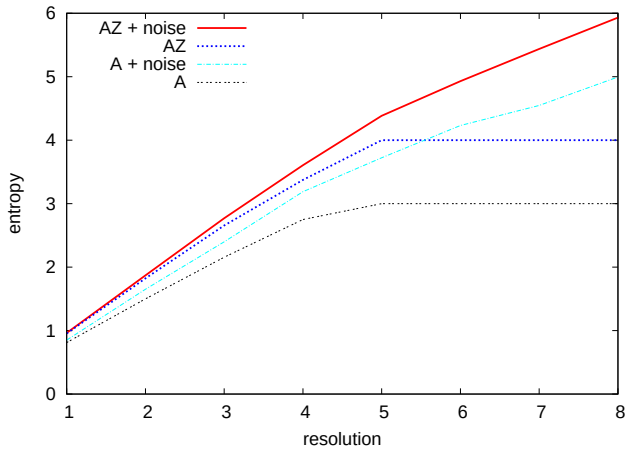


(b) Entropy-Resolution signature (at scale 1).

Figure 5. Signatures of simple strings: The first pattern is a set of alternating 1s and 0s. The second pattern consists of sets of six 1s separated by a varying number of 0s. Adding random noise the strings raises their entropy, but the shape of their signature is preserved.



(a) Entropy-Scale Signature (resolution $r = 8$)



(b) Entropy-Resolution Signature (at scale 1)

Figure 6. Signatures of Character Patterns: The patterns are composed of binary representation of 8-bit ascii characters. The first pattern is a set of the repeating character 'A.' The second pattern consists of repeating the string 'AZ.' Adding random noise the strings raises their entropy, but the shape of their signature is preserved.

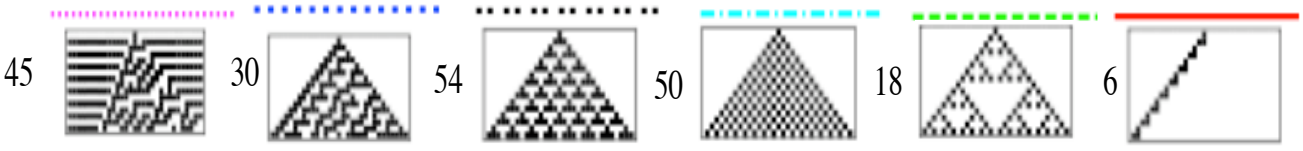
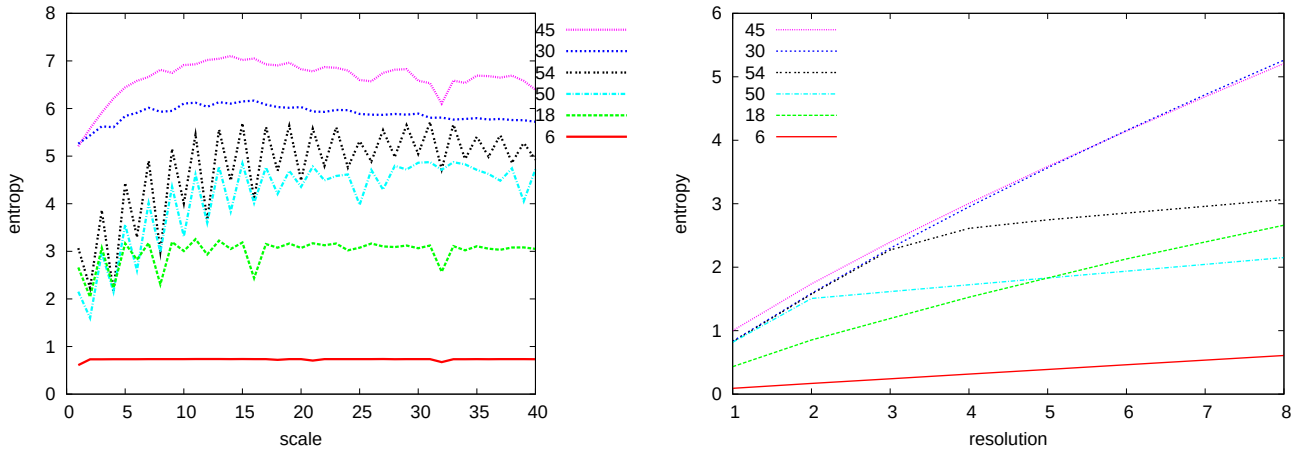


Figure 7. Six Sample eight-bit Cellular Automata. The Cellular Automata evolve vertically, with each line representing a successive time step.



(a) Entropy-Scale signature (resolution $r = 8$).

(b) Entropy-Resolution signature (at scale 1).

Figure 8. Signatures of cellular automata: These figures show the signatures for six of the possible 256 8-bit cellular automata. The image of the cellular automata is shown in the middle of this figure. Note that cellular automata that look similar (such as 45 and 30, or 54 and 50) have similar looking signatures. The signatures also reflect the complexity of the automata, and in the case of cellular automata 18, the signature reflects its fractal property.

Figure 8 shows the signatures for these six CAs. CAs 45 and 30 are chaotic and have similar high entropy signatures that are not significantly influenced by scale. CAs 54 and 50 both have simple repeating patterns and both have similar signatures that oscillate heavily with scale. CA 18 has a fractal pattern, which is captured by the spikes in its signature occurring at powers of two. CA 6 has a very simple pattern, which is captured in a low entropy signature.

Mixed Strings

In many cases the data we are analyzing may be composed of data from many different sources that we do not have access to directly. A desirable characteristic of the mixed string signature is that it retains the salient characteristics of its component strings. This property is especially important when we need to detect a change in the properties of an individual source. Figure 9(a) shows how the ES signature preserves the properties of the separate data sources. In this figure, a string is created by interleaving the bits from two other strings. Every odd bit comes from the first string and every even bit comes from the second string. Note the signature of the mixed string shares the structure of its constituent strings, rather than appearing as a random string. Furthermore, when a string with a pattern is interleaved with a random string, the full signature is translated up, but still contains the pertinent information of the initial string. Note that half the bits in this string are random, yet the structure of the initial string is still preserved.

5. Software Signatures

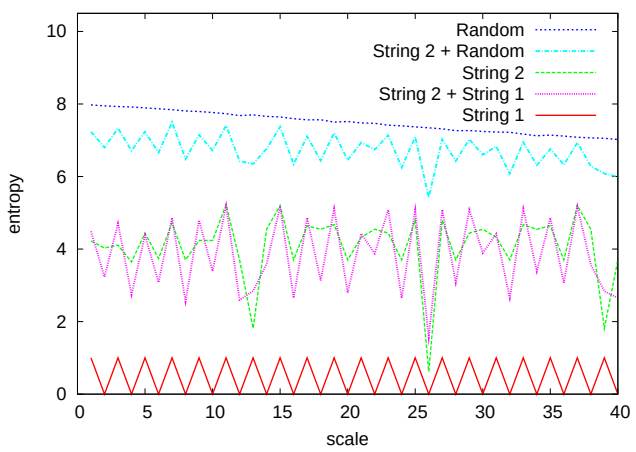
In the previous sections we showed how complexity signatures can be of value by illustrating their use on a set of artificially generated data strings. In this section, we examine empirically how those signatures work on real data obtained from software packages. This analysis is performed by first converting a call-graph representation of the software into a data string using a specialized traversal. Then, the entropy based signatures are used to assess the complexity of the resulting data string. The complexity measure is then used to predict the number of revisions that were made to each source code file. In this instance the number of revision counts is used as a risk marker, since it is directly correlated with cost overruns, delivery delays, and ultimately, failures.

Converting Graphs into Data Strings

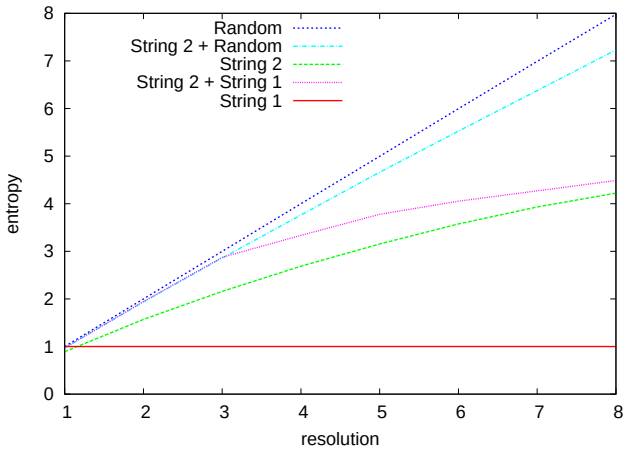
The first step in our approach to analyzing the software call-graph is to create a representation of the call-graph in the form of a data string while conserving its structure. We do this by making traversals through the call-graph and concatenating each node visited to the end of the data string. The goal in our traversal is to approximate the calls that would be made by a running program. This traversal is done through a modified random walk where each node that is visited is marked. If a loop is detected (a node is visited which is marked), then the walk direction is reversed simulating a return from a function call. An example traversal is shown in Figure 10.

Performance on Software Data

We tested how well a complexity analysis of a software's call graph can be used to predict the number of revisions that were made to each code file in a software package. We used five java-based software packages from the open source commu-



(a) Entropy-Scale signature (resolution $r = 8$).



(b) Entropy-Resolution signature (at scale 1).

Figure 9. Signatures of mixed strings: The string "String 2 + String 1" is composed of interleaving the bits of String 1 and String 2. Note that the signature of this interleaved string contains properties of both Strings 1 and 2. When String 2 becomes interleaved with a random string, its signature becomes much different.

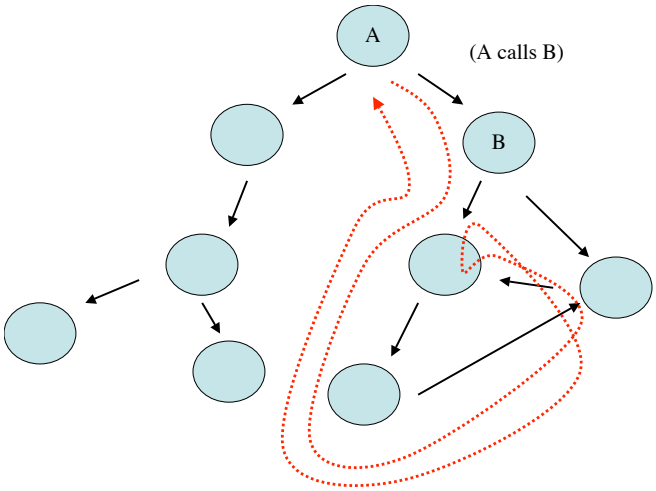


Figure 10. Traversal heuristic: A random walk is performed on the graph until a node is revisited. Then the walk is reversed to simulate the returning from a function call.

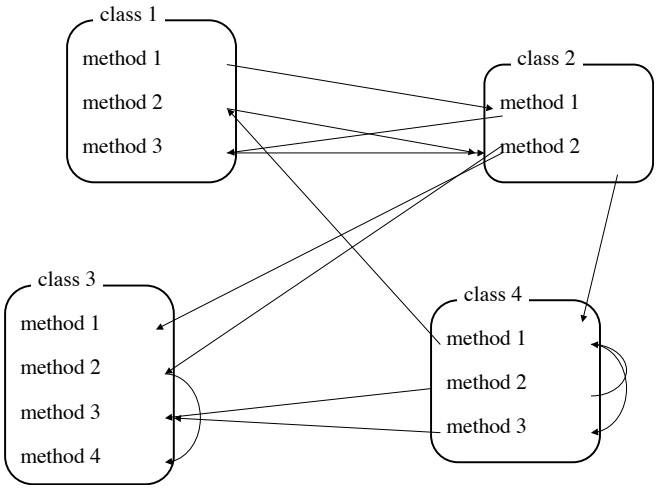
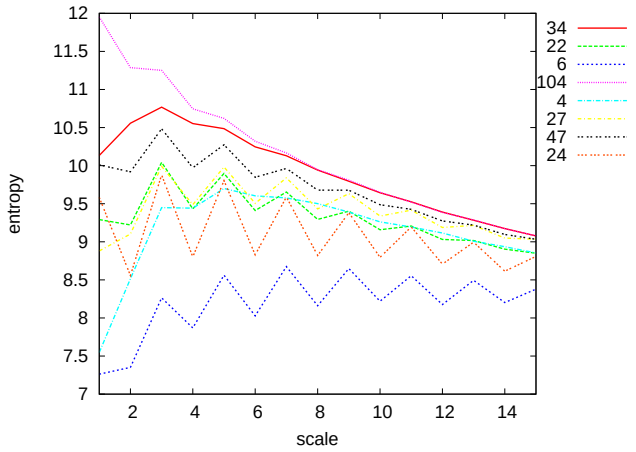


Figure 11. **Call Graph Between Methods**. Each class can contain several methods. The directed call graph is defined by the calls between methods.

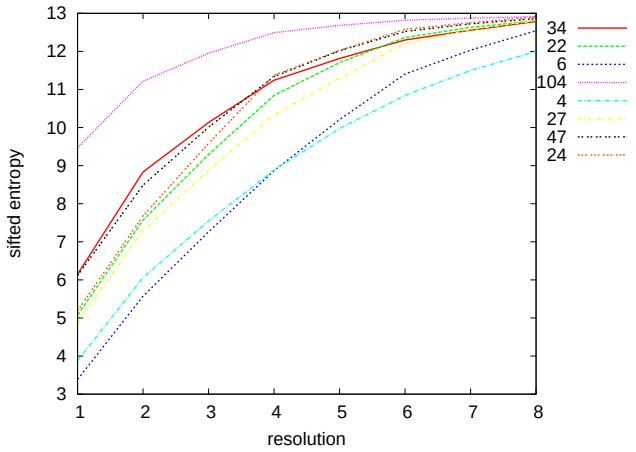
nity[2]. These packages contained a total of 300,000 lines of code. The software was composed of java code files, with typically one class per file. Most classes contain methods (functions), which can be called from other methods within the class or from methods outside of the class. For each software package, a call-graph was generated between all of the methods as shown in Figure 11.

For each class, a data string was generated by taking a series of traversals where each traversal started at a random method within the class. All the traversals were then concatenated together to form the data string representation of the call-graph. Figure 12 shows the signatures for the eight largest methods, where each method is labeled by the number of revisions it received. A close look at Figure 12(b) shows two different levels of clusters. For small resolutions ($r = 1$ and $r = 2$), there are four clusters. The first has the methods with 4 and 6 revisions, the second has the methods with 22, 24 and 27 revisions, the third has methods with 34 and 47 revisions and the last has the method with 104 revisions. At resolutions of $r = 3$ and higher, the two middle clusters merge together. The discriminating capability of this method is remarkable considering it is based on generating data strings from the

[2]The packages were "Pmd," "Megamek," "Jedit," "Tyrant," and "Pdfbox." They came from sourceforge.com and were chosen from active software packages that contained revision counts in their CVS tree.



(a) Entropy-Resolution signature (at scale 1).



(b) Entropy-Scale signature (resolution $r = 8$).

Figure 12. System signatures for software packages: Each curve refers to a stream generated from a single class file. The key shows the number of revisions for the class file. Strings with higher entropy tend to have higher revision counts.

software call graph and then obtaining the entropy of that string at different resolutions. No information about the graph is used or needed to cluster the methods. Figure 12(a) shows the scale dependence of each method at resolution $r = 8$. Though the method requiring 104 and 6 revisions are singled out, the rest of the methods share similar signatures.

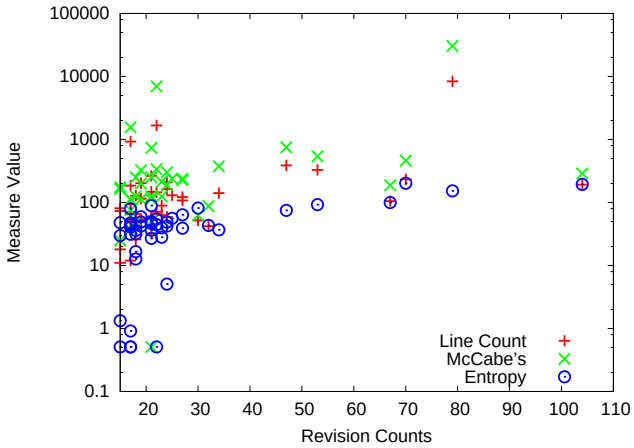


Figure 13. Effectiveness of entropy-based signatures: Each data point of a given color represents a class. The entropy based measure is a significantly more accurate predictor of the number of revisions made to a class than McCabe's Complexity.

In order to compare these results to currently used methods for predicting revision counts, we have isolated a cross section of Figure 12(b). Unfortunately, the overall quality of the data was poor as many of the classes did not have their revision counts properly recorded. Therefore we exclude large classes that had almost zero revision counts, because the revisions were never entered into the system.

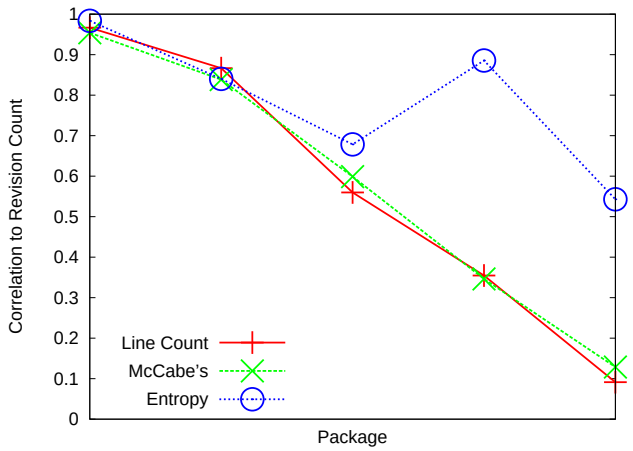
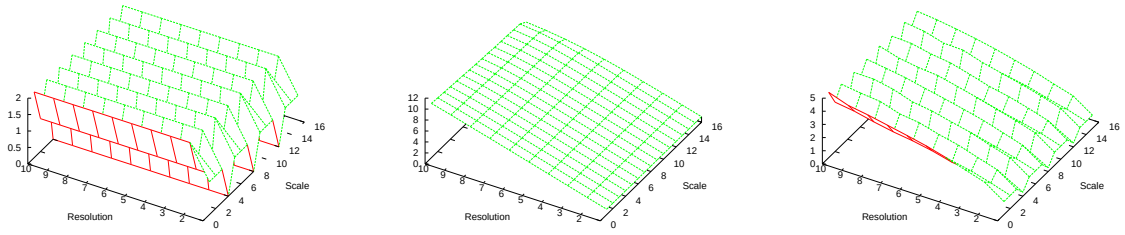


Figure 14. Correlation of system signatures to revision counts. The entropy based measure is significantly more correlated to revision counts for the classes with the ten-most revisions.

Figure 13 shows the performance of the entropy measure at $r = 3$, $s = 1$ vs. McCabe's Cyclometric Complexity, a widely used software complexity metric [9]. McCabe's Complexity measures the number of times that the flow is broken in a software file, including flow breaks coming from method calls and "if" statements. The entropy based signature is nearly linearly correlated with revision counts whereas the McCabe's measure shows little correlation with the revision counts. Note, the difference in predictive ability becomes more pronounced when the revision counts are higher, a regime where the quality of the data is more reliable.

In Figure 14, we extend this result to the full five software packages and include a simple line count as a third indicator



(a) String "1100110011.."

(b) Uniformly random pattern.

(c) String "1100110011.." + Uniformly Random

Figure 15. Full 3D Entropy-Resolution-Scale signatures. The third signature is the superposition of the first two strings, and both systems' characteristics (slope for random and peak/valley for regular string are present.

of software complexity. In this figure we show, using Pearson's Correlation, the predictive power of each of the measures for the ten classes of each package that had the highest revision counts. The results show that the entropy-based measure performed significantly better than the other metrics on two of the packages, slightly better on a third package and about equally well on the remaining two packages.

6. Discussion and Future Work

In this work we have shown that system level signatures based on the changes in entropy when a system is viewed at different resolution and scales are effective in capturing salient properties of the system. These signatures have both general system markers to distinguish between different types of behavior in a system and detailed system markers that are sensitive to perturbations in the system. The latter in particular is crucial in using such signatures as aids in health monitoring.

In both artificial data strings and simple discrete systems, these signatures provided good discriminating capabilities. Furthermore, when applied to a generic set of software packages, they proved more effective in predicting failures than standard "size" measurements for software, including McCabe's Cyclometric Complexity. This is an encouraging result as these signatures were not tuned to the domain. We anticipate stronger results when we apply these signatures to richer (and cleaner) data.

In the experiments reported in this article, we have used the Entropy-Scale and Entropy-Resolution signatures as independent signatures. Our current work focuses on binding the two together to provide 3D system signatures. Figure 15 shows a sample of such signatures on a simple repeating pattern, a random pattern and a pattern where the two have been combined. The 3D signature of the mixed signal clearly shows the dominant structure of both component systems in the form of the repeating valleys and the overall slope. We are also investigating more expressive visualization methods, including color-intensity 2D charts instead of 3D charts. We are also extending this work into telemetry data composed of real numbers and are addressing the third deficiency of Entropy as a complexity measure described in Section 3 by generalizing Entropy to account for proximity in the data. Finally, we are looking into applying these signature to real aerospace systems (e.g., shuttle engines) to demonstrate their health monitoring properties in engineered complex systems.

References

[1] Y. Bar-Yam, editor. *The Dynamics of Complex Systems*. Pegasus, 1997.

[2] D. A. Bearden. Observations on complexity and costs for over three decades of communications satellites. In *53rd International Astronautical Congress of the Inte rnational Aeronautical Federation*, Houston, TX, October 2002. 02-IAA.1.4.01.

[3] T. Bedford and R. Cooke. *Probabilistic Risk Analysis: Foundations and Methods*. Cambridge University Press, Cambridge, UK, 2001.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[5] Codd E. F. *Cellular Automata*. Academic Press, New York, 1968.

[6] M. Gell-Mann and S. Lloyd. Information measures, effective complexity and total information. *Complexity*, 2:44–52, 1996.

[7] S. A. Kauffman. *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, 1995.

[8] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1997. 2nd Edition.

[9] T. J. McCabe and A. H. Watson. Software complexity. *Crosstalk, Journal of Defense Software Engineering*, 7(12):5–9, December 1994.

[10] D. A. Meyer. Toward the global: Complexity, topology and chaos in modelling simulation and computa-

tion. In Y. Bar-Yam, editor, *Unifying Themese in Complex Systems, Volume 1 (Proceedings of the First International Conference on Complex Systems)*, pages 343–355. 2000.

[11] Shalizi C. R. Methods and techniques of complex system science: An overview. In *Complex Systems Science in Biomedicine*. Kluwer, 2004.

[12] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

[13] J.S. Shiner, M. Davison, and P.T. Landsberg. Simple measure of complexity. *Physical Review E*, 59:1459–1464–283, 1999.

[14] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[15] S. Wolfram. *Cellular Automata and Complexity – Collected Papers*. Addison Wesley, 1994.

[16] D. H. Wolpert and W. G. Macready. Self-dissimilarity as a high dimensional complexity measure. In Y. Bar-Yam, editor, *Proceedings of the Fifth International Conference on Complex Systems, 2004*, 2005.

Kagan Tumer *leads the "sensing and control in distributed adaptive systems" group at the computational sciences division of NASA Ames Research Center. He has authored over sixty publications; is an associate editor of the "Pattern Recognition" journal (Elsevier) and the "complex systems and inter-disciplinary science" book series (World Scientific); and has chaired numerous workshops/symposia. His edited book titled "Collectives and the Design of Complex Systems" was released by Springer in 2004.*

Adrian Agogino *is a researcher at the University of California Affiliated Research Center at NASA Ames Research Center. His interests include complexity, multi-agent coordination, reinforcement learning and visualization. He has authored a dozen peer reviewed publications in these areas.*